NTNU – Trondheim
Norwegian University of
Science and Technology

Title

Vegard Stjerna Lindrup

Submission date:	May 2016
Responsible professor:	Tor Engebret Onshus
Supervisor:	

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Title:	Title
Student:	Vegard Stjerna Lindrup

Problem description:

Dette legges til i DAIM, og blir derfor fjernet før innlevering.

Responsible professor:	Tor Engebret Onshus
Supervisor:	

Abstract

Mobile robot platforms etc...

Sammendrag

Mobile robotplatformer kan kjøre rundt og...

Preface

Hva synes jeg om oppgaven? Kjempeartig! Eget acknowledgemet-kapittel?

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence

AIV Autonomous Inspection Vehicle

CLM Concurrent Localization and Mapping

CP Cathodic Protection

DRC DARPA Robotics Challenge

GUI Graphical User Interface

HMI Human-Machine Interaction

HSE Health, Safety and Environment

IFR International Federation of Robotics

ISS International Space Station

LIDAR LIght Detection And Ranging

MIMROex Mobile Inspection and Monitoring Robot, experimental

NDT Non-destructive Testing

MIT Massachusetts Institute of Technology

NUI Natural user interface

OCS Operator Control Station

PR Personal Robot

ROS Robot Operating System

ROV Remotely Operated Vehicle

RPAS Remotely Piloted Aerial System

RTAB-Map	Real-Time Appearance-Based Mapping
SDF	Simulation Description Format
SIFT	Scale-invariant feature transform
SLAM	Simultaneous Localization And Mapping
SSD	Solid State Drive
SURF	Speeded Up Robust Features
STAIR	Stanford AI Robot
UAV	Unmanned Aerial Vehicle
URDF	Unified Robot Description Format

List of Algorithms

Chapter 1
Introduction

Introduction

1.1 About the Thesis

1.1.1 Future Goal (The final product)

A nice description of a potential final product.

1.2 Implementation Overview

1.3 Thesis Structure

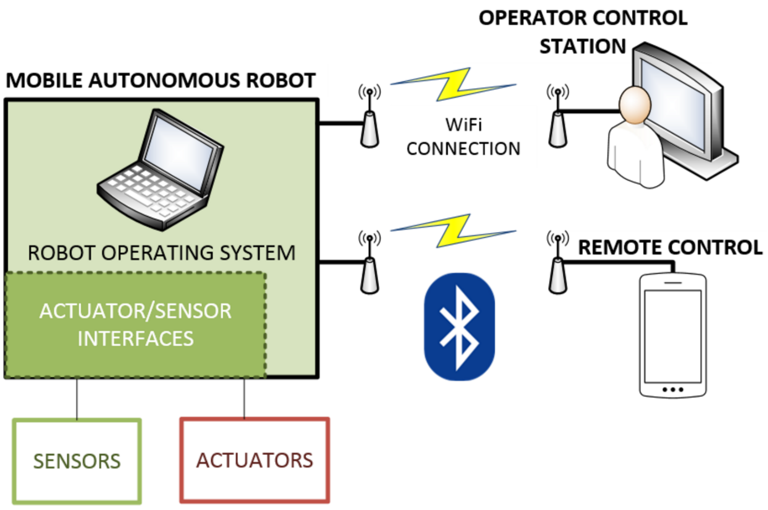


Figure 1.1: System Concept.

Chapter 2

Task Outline and Planning

2.1 Introduction

The original problem description given at the beginning of this semester is very open. A significant part of the project is oriented towards

2.2 Task definition

2.3 Specification

2.4 Planning

2.4.1 Work Breakdown Structure

Chapter 3
Background Theory

3.1 Robotic Maintenance of Industrial Installations

3.1.1 Introduction

This project is a small step towards a larger long-term goal concerning robotic maintenance. This section puts the following background theory, and the implementation described in chapter 4, into the context of automated robotic maintenance on industrial installations. It is important to describe how maintenance and inspection of industrial installations is done today, before application of robotic maintenance is discussed.

3.1.2 Potential Maintenance Tasks

Hidden failure modes: PFD: What is the probability that a device (Fire detector, shut down valve, etc.) will fail when needed? Solution: Periodic maintenance.

3.1.3 Offshore Installations

Corrosion

Offshore installations are regularly, if not continuously, exposed to harsh weather conditions in the form of wind and seawater. Presence of seawater, either through direct contact or in the form of drops and vapor, forms a very corrosive environment. The offshore and marine environment is classified as the most corrosive environment in ISO 12944[ER12]. It is essential to provide countermeasures to ensure safe and reliable operation over the lifetime of the installation. Common corrosion prevention methods are[ER12]:

– Sacrificial Anodes.

– Cathodic Protection (CP) in the form of a DC-current.

– Protective coating.

In terms of maintenance, the sacrificial anodes can be subjected to periodic inspections and replacements, which could be done a robot. CP can more easily be implemented with automated self tests, and should normally not require any inspections and maintenance[ER12]. Application of protective coating should ideally be applied in the controlled environment of a workshop. If protective coating is to be applied at sea, one should strive to make the conditions as favorable as possible.

Fatigue

Waves, wind, water currents and other forces subject offshore installations to structural stress.

3.2 Robotic Maintenance Today

Gas leak detection: [SWB$^+$14]. DARPA robotic challenge. Industrial ROS.

Trends and Potential

The typical pre-programmed assembly robots still dominate the robotic market. They are usually found in manufacturing plants and large scale production facilities[ifr], e.g. the automotive industry, where they perform dull, tedious tasks much faster and with higher accuracy than people. A notable trend in modern robotics is increased human-robot collaboration[Bog16]. Many new robots are being build for the human workspace, both in terms of safety and collaborative functionality. This trend is a step along the way of moving robots out of the controlled environment of a factory floor, and into the real world where a high degree of autonomy is required.

A report by Metra Martech[GC11], a market research firm referenced to by International Federation of Robotics (IFR)[1], points to three areas with a high potential for robotic applications:

– Dangerous jobs, e.g. handling dangerous materials or work in high risk environments.

– Jobs that are economically infeasible in a high wage economy.

– Work which is impossible or highly inconvenient for humans, e.g. space exploration, subsea maintenance or assembly of heavy components.

[1]http://www.ifr.org/robots-create-jobs/

All of these factors motivate the development of robots for autonomous robotic maintenance.

Subsea Maintenance and Inspection

Subsea maintenance is perhaps the field that have seen the greatest advancements in autonomous inspection and maintenance. As offshore installations are moved to the seabed, maintenance and inspection has become a significant challenge. This has resulted in a widespread use of Remotely Operated Vehicles (ROVs). Recent developments in other fields, e.g. computer vision, human-robot collaboration and machine learning, has resulted in new Autonomous Inspection Vehicles (AIVs) and Autonomous Underwater Vehicles (AUVs) capable of performing inspection and simple maintenance tasks autonomously[JWA+12][RCR+15]. A driving factor behind the transition from ROVs to AUVs is cost reduction through increased offshore campaign efficiency.

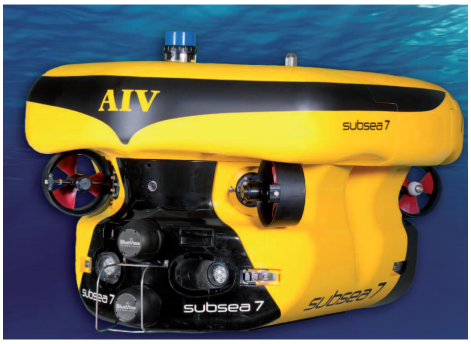


Figure 3.1: Subsea 7's AIV. This is the first commercial autonomous inspection vehicle for subsea operations [pre]

Disaster Responce

Robots in disaster response, relief and recovery solve many of the same problems faced by maintenance robots. Disasters, such as the Tsunami which struck Japan in 2011, has proved to be particularly demanding for robots, both in terms og technical difficulties as well as the process of deployment. Many of the robots which were deployed at Fukushima were already aging, and the operators had to receive training before deployment, thus increasing the response time[KFO12]. A paper from Japan Atomic Energy Agency[KFO12] highlights how the lack of stakeholder involvement could have been the cause

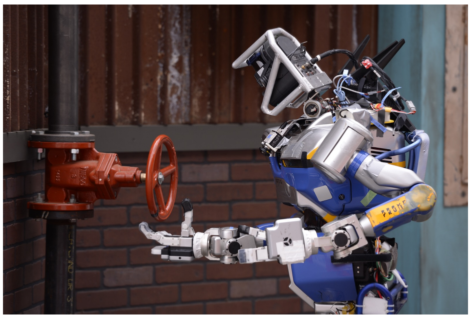


Figure 3.2: Team HRP2-Tokyo's robot turning a valve during DARPA Robotics Challenge 2015 (Image credits: DARPA Robotics Challenge)

of long response times. The same paper points out that the robots were developed for the sake of development, and not with emergency response as the main purpose[KFO12].

DARPA Robotics Challenge (DRC)[DRC] was launched in response to the Fukushima disaster of 2011. The purpose of the competition is to accelerate innovation, research and development in robotics for disaster response in cases where humans cannot operate. Some of the tasks the competitors faces in 2015 include valve turning, traversing rubble and driving a vehicle through a course before egressing out of the vehicle.

Robotic Maintencance of Process Plants

Robotics has several potential applications in topside process plants, and particularly in remote oil and gas installations. In light of the factors listed earlier, maintenance and service robots will be designed with the following goals in mind[KMP15]:

– **HSE** - Reduce risk exposure for personnel and environment.

– **Efficiency** - Accomplish more with less effort, resources and time. This means cost reduction by keeping downtime to a minimum with the least amount of effort.

Autonomous and teleoperated inspection and maintenance today is usually only found at subsea installations. Topside installations on the other hand are still maintained and inspected manually, with some notable exceptions. Small Unmanned Aerial Vehicles (UAVs) or Remotely Piloted Aerial Systems (RPAS) have become commonplace over the last decade. On topside installations, they are being used for visual inspection of inaccessible structural parts such as flare stacks or the exterior of oil rigs.

Fraunhofer Institute for Manufacturing Engineering and Automation[2] has developed a robot, called Mobile Inspection and Monitoring Robot, experimental (MIMROex), with capabilities which are quite similar to the prototype used during the work on this thesis. MIMROex is equipped with a camera for visual inspections as well as microphones, vibration and sensors for fire and gas detection. It is also certifiable in accordance with the explosion protection standard IEC 60079[MIM].

[2]http://www.ipa.fraunhofer.de/en.html

3.3 Modelling and Simulation

3.3.1 Some Terminology

Coordinate Systems and Poses

Robot Joints

All links are connected to each other by joints.

Coordinate systems are essential in the field of robotics.

3.3.2 Robot Modelling

3.3.3 Simulating in Gazebo

3.4 ROS

3.4.1 Introduction

The Robot Operating System (ROS) is a collection of software libraries, tools and drivers intended for robot software development. A ROS installation can be tailored to meet the demands of a wide range of robots with varying complexity. ROS is usually installed in the form of an already built Debian-package. These packages are only compatible with a few versions of Ubuntu which are specified on the ROS homepage. When installed and configured, ROS will run on top of Linux, and can be perceived as and extention of Linux itself. Installing ROS from source is possible, but not recommended [ROSb].

Roots of ROS can be traced back to Stanford University at the beginning of the 2000s. At Stanford, several robotics software frameworks, including Stanford AI Robot (STAIR) and the Personal Robot (PR) program, were created to provide dynamic, flexible and well tested foundations for further robot development and research. In 2007, a nearby start-up company and robot incubator, Willow Garage, sought to build upon these concepts, and initiated a collaborative and open development process of a new software framework. This framework eventually became ROS[ROSa][QGS15]. The framework can be used under the BSD open-source license, which means that ...[**?**
] Today, ROS comes in many forms and comprise hundreds of advanced packages, algorithms and drivers, making it applicable for hobbyists, industrial automation, research and everything in between.

3.4.2 Important ROS Concepts

The following descriptions are included in order to provide a complete, self-contained description of the project implementation. Similar descriptions can be found on the official ROS website[3], as well as in any book on ROS (for example [QGS15]).

The ROS Graph

A ROS system comprise a set of small programs that communicate with each other through messages. These programs become nodes in the ROS graph. The nodes communicate with each other by publishing and subscribing to topics that form the edges of the graph. A topic must have the format of one of the specific data types provided by ROS. For example, a node which receives temperature data from a thermometer, may publish the data as a topic on the ROS system with the type `sensor_msgs/Temperature`. There are many other data formats, e.g. velocity messages, `geometry_msgs/Twist`; images, `sensor_msgs/Image`; odometry messages, `nav_msgs/Odometry` and so on. Each node in the graph are typically POSIX processes, and the edges are TCP connections[QGS15].

roscore

`roscore` is an essensial part of any ROS system as it enables nodes to communicate with each other. When a node is started, it will inform `roscore` of which topics it publishes and which topics it wish to subscribe to. Then, `roscore` will provide the information which allows the node to form a peer-to-peer connection to other nodes.

Project Structure and catkin

A ROS project will usually utilize the catkin build system.

3.4.3 ROS-Related Tools

Robot Modelling In URDF

Visialization in RVIZ

Simulation in Gazebo

3.4.4 Structure of a ROS Application

3.4.5 Notable Robots Running ROS

PR2 - Personal Robot 2 PR2 is one of the first robots designed to run ROS [QGS15], and also one of the most advanced and capable robots with ROS today.

TurtleBot TurtleBot is a cheaper ROS-ready alternative to PR2.

[3]http://www.ros.org/

Robonaut 2 Robonaut 2, a dexterous humanoid robot, currently resides within the International Space Station (ISS) 400 km above the earth's surface. In 2014, a SpaceX Dracon capsule brought ROS as well as a pair of legs for Robonaut up to the ISS[ROSc]. Robonaut is designed for research on human-robot collaboration in space, and human-like tasks. For more information, follow **this link**[4] to a talk on ROS in space from ROSCon 2014.

Being the first robot with ROS to be launched into space,

Example of an industrial robot with ROS goes here! TODO!!!!!!!!!!

3.5 Software

3.5.1 Qt

3.5.2 PCL

3.6 The Kinect Sensor

3.7 Software Tools

3.7.1 Point Cloud Library

3.7.2 ROS

3.7.3 Qt

3.7.4 Current Research and Applications

3.8 Introduction to Sensors in Autonomous Robots

3.8.1 Depth Cameras

Different Methods for Depth Perception

In the context of this thesis, a depth camera is considered to be a sensor which the functionality of a regular video camera combined with the ability to perceive a depth image.

A depth camera can be described as a regular color video camera with the ability to create spatial images. In the context of this thesis, a depth camera can more precisely be described as a RGB-D camera, which is short for red, green, blue and depth camera. A regular RGB camera will project a spatial scene onto a rectangular pixel grid, where each pixel contains intensity values for red, green and blue colors.

[4]https://vimeo.com/106993914

These pixel values represents the detected scene. A major problem with RGB cameras is the significant loss of information. The information loss is mostly a consequence of 3d to 2d projection and digital quantization. RGB-D cameras have the means to reduce this information loss by mapping the pixel values to spatial coordinates, turning each pixel into voxels and the image into a point cloud of voxels.

Different variations of depth cameras will usually fall into one of two categories: active or passive. Passive sensors perceive the surroundings as it is, without actively interfering with the environment as a part of the sensing process. A typical passive RGB-D sensor is the stereo camera. Stereo cameras use a stream of synchronized image pairs to perceive depth. The image pairs are displaced along the horizontal axis, and the depth information is extracted by searching for mutual information in the image pairs. How far the information is displaced from the left to the right image is directly related to how far away from the camera the information source is located.

Active sensors depend on some form of projection onto the surroundings. For depth cameras, the projection is usually in the form of laser or infra red light. In RGB-D cameras it is essential that the projected light is distinguishable from the visible spectrum. The Kinect sensor used in this project is an example of an active RGB-D sensor. A proper introduction to the Kinect, will follow shortly.

Natural User Interfaces - Origin of the Kinect

Forslag 1: When a group of designers are developing a new Graphical User Interface (GUI), they will often use a conceptual model when planning their design. The conceptual model is the mental model the designers want to put into the head of the user. All users will develop their own individual mental model, which is their high level understanding of how the GUI works. A conceptual model may contain metaphors for things the user already is familiar with. A painting program for example may use a metaphor for a canvas, paint brushes and palettes. When the mouse icon changes to a paint brush, most users will have an intuitive understanding of what it can do and how it is used. A Natural user interface (NUI) will seek to remove the metaphors and create a more seamless interaction between the user and the machine. Some NUIs may allow a user to write text with a pen instead of a keyboard, or dictate a letter with their voice while the computer converts audio into text.

Forslag 2: The idea behind a NUI is to make Human-Machine Interaction (HMI) as seamless and natural as possible. A NUI allows the user to communicate without tools such as a keyboard or a mouse. For decades, NUIs have only existed as ideas, science fiction or research projects. This has changed dramatically over the last ten years, and NUIs can now be considered to be ubiquitous. Today, the most common form of NUIs is the touch screen found in smart phones and tablets.

The Microsoft Kinect sensor was initially designed as a NUI for the Xbox 360 gaming console. The sensor allows users to use gestures and sounds to play console games. Later on, Microsoft has released SDKs, enabling developers to create NUI applications for for Windows.

The modern RGB-D sensors which are commonly used in robot research projects today were initially intended as NUI.

Kinect for Xbox 360

Kinect for Xbox 360 is the RGB-D sensor used in this project. The device was initially intended as a NUI for gaming and office applications. Possible use cases were inspired by early NUI research at Massachusetts Institute of Technology (MIT) and, later on, the science fiction movie Minority Report, where Tom Cruice interacts with a computer by using hand gestures [WA12]. The Kinect sensor is equipped with a depth sensor, a regular color camera, a microphone array and a tilt motor. The color camera in combination with the depth sensor forms what is usually referred to as a rgb-d sensor, i.e. a combined color and depth camera. This feature, combined with the relaticely low cost and accessability of the sensor as contributed to make the Kinect very popular in research projects related to Simultaneous Localization And Mapping (SLAM) and robotics.

Today, the the Kinect for Xbox 360 has been succeded by the Kinect for Xbox One, and is now considered to be a legacy device. Those considering to use the legacy Kinect should be aware of that it is becoming increasingly difficult, if not already impossible, to get hold of a new Kinect for Xbox 360.

3.8.2 Plannar Laser Sensors (LIDAR)

A plannar laser sensor, known as e.g. laser proximity sensors or laser radars, can all be referred to as LIDARs.

Scanning Laser Range Finder, URG-04LX-UG01

3.8.3 Odometers

3.8.4 Sensor Fusion

3.9 Simultanious Localization and Mapping (SLAM)

3.9.1 Introdunction to SLAM

SLAM, also known as Concurrent Localization and Mapping (CLM), is a class of solutions to the problem of determining an agents location and pose in an unknown environment, while simultaneously mapping the same environment.

3.9.2 Hector SLAM

3.9.3 RTAB-Map

Real-Time Appearance-Based Mapping (RTAB-Map) is a graph-based SLAM system intended to handle the ”kidnapped robot-problem” as well as multi-session mapping[LM14]. Both these problems become relevant whenever a robot is shut down and moved to a new unknown location in the same area. RTAB-Map is the core feature in the implementation that is described in this thesis. Some factors which motivated the use of RTAB-Map are:

– It is a SLAM method which requires an RGB-D sensor, for example a Kinect.

– RTAB-Map has a ROS wrapper, `rtabmap_ros`, which eases the process of integrating it with the mobile robot.

– It includes obstacle detection in 3d.

– It can use **SURF!** (**SURF!**) or **SIFT!** (**SIFT!**) from OpenCV for object recognition.

– It has a long-term memory intended for large scale multi-session mapping.

3.9.4 Octomap

3.10 Navigation

Figure 3.3: Awesome Image

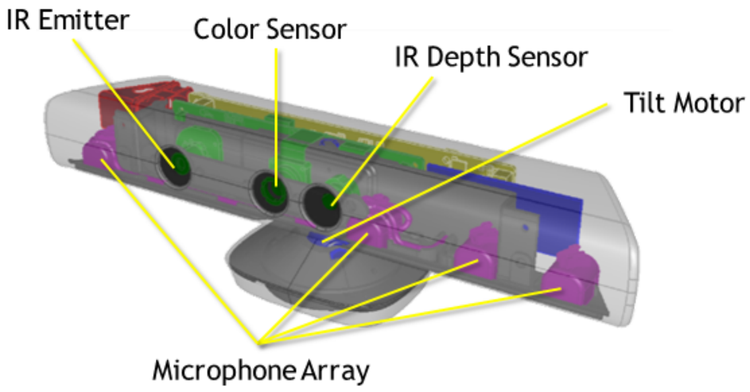


Figure 3.4: Awesome Image

Chapter 4
Implementation

Implementation procedure for mobile robot:

- Decide on ROS message interface.
- Write interfaces for the motor drivers.
- Create a description of the physical structure and properties of the robot in Unified Robot Description Format (URDF).
- Extend the model to enable simulation in Gazebo.
- Publish coordinate transform data via *tf* and visualize it in rviz.
- Add sensors, with driver and simulation support.
- Apply algorithms for navigation and other functionality.

4.1 Modeling

4.1.1 Physical Dimensions

The inertia tensor:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{4.1}$$

Inertia tensor for a solid, uniform cylinder where the radius r is measured in parallel to the $x - y$ plane, and h is parallel to the z axis:

$$I_{cylinder} = \frac{1}{12}m \begin{bmatrix} (3r^2 + h^2) & 0 & 0 \\ 0 & (3r^2 + h^2) & 0 \\ 0 & 0 & r^2 \end{bmatrix} \tag{4.2}$$

Inertia tensor for a solid, uniform cuboid. The subscript of l indicates which axis l is measured along:

$$I_{cuboid} = \frac{1}{12}m \begin{bmatrix} (l_y^2 + l_z^2) & 0 & 0 \\ 0 & (l_x^2 + l_z^2) & 0 \\ 0 & 0 & (l_x^2 + l_y^2) \end{bmatrix} \tag{4.3}$$

4.1.2 Coordinate Frames

4.2 Simulations

4.3 ROS Nodes for Motion Control

4.3.1 Velocity Command Sources

There are four ways to control the robot:

- Local keyboard input.
- Wireless teleoperation from the Operator Control Station (OCS).
- Wireless teleoperation from a handheld Bluetooth device.
- Commands from the navigation stack in ROS.

4.4 Operator Control Station (OCS)

The OCS allows an operator to control and monitor the robot through a graphical user interface. `MainWindow`

4.4.1 Graphical User Interface

A Qt-based GUI...

4.5 The Handheld Remote Control - "Robot Leash"

Because the OCS is only partially implemented, an operator will not have access to all the features on the robot. In addition, as a safety precaution a person should be close to the robot at all times, and be ready to pull the plug. Furthermore, it is hard to control a moving robot through the on-board keyboard. These problems were countered by the Android-based remote control, ”Robot Leash”.

4.5.1 Connecting to a the Robot

1. The first screen after scanning for devices. There is no device filtering, and the user can select any device, but only connect through a specific service.
2. After selecting a device which provides the correct service, the user will be prompted to pair the devices.
3. The smartphone and the robot is now paired, and velocity commands from the blue control stick are passed to the robot via Bluetooth.

http://developer.samsung.com/technical-doc/view.do?v=T000000117

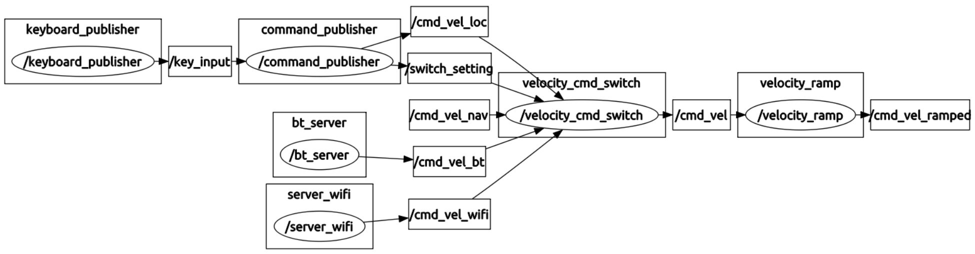


Figure 4.1: Nodes and topics for motion control.

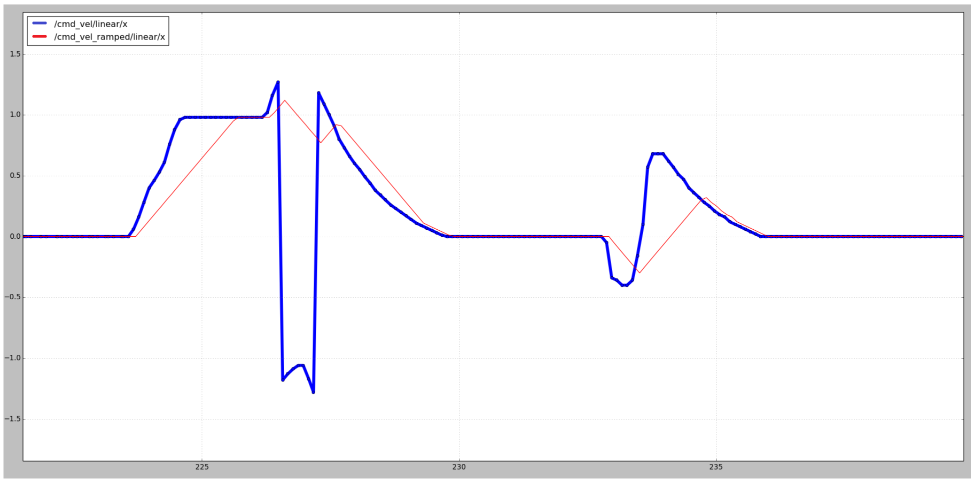
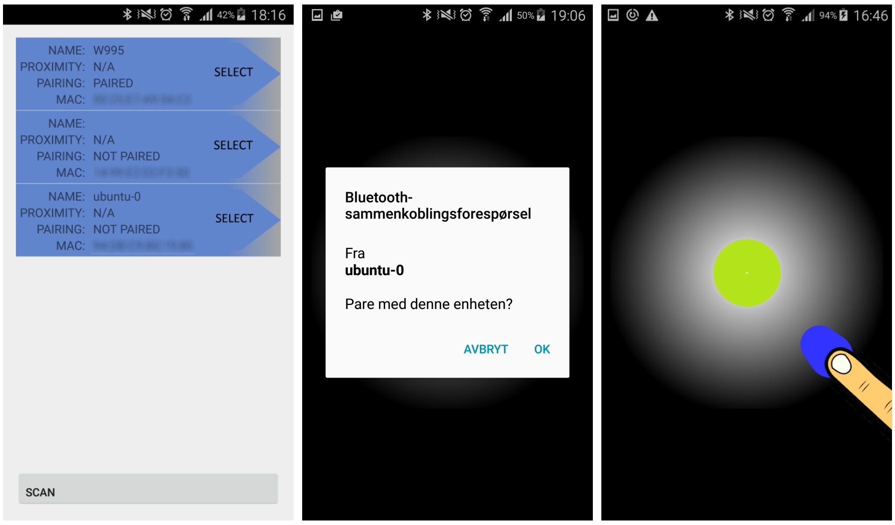


Figure 4.2: Velocity command ramping. The blue line represents commands entering "`velocity_ramp`", while the red line shows the acceleration constrained output command.



NAME: W995
PROXIMITY: N/A
PAIRING: PAIRED
MAC: [redacted]

NAME:
PROXIMITY: N/A
PAIRING: NOT PAIRED
MAC: [redacted]

NAME: ubuntu-0
PROXIMITY: N/A
PAIRING: NOT PAIRED
MAC: [redacted]

Bluetooth-sammenkoblingsforespørsel

Fra
ubuntu-0

Pare med denne enheten?

(a) First activity with device list.

(b) The user is prompted to pair with the robot.

(c) Controlling the robot with the stick.

Figure 4.3: A typical use case for ”Robot Leash”.

Chapter 5 Testing

5.1 Testplan

5.2 Results

5.2.1 Simulations

5.2.2 Live Testing

Safety Features

Multi Session Mapping

Navigating an Obstacle Course

Dynamic Replanning

5.3 Discussion

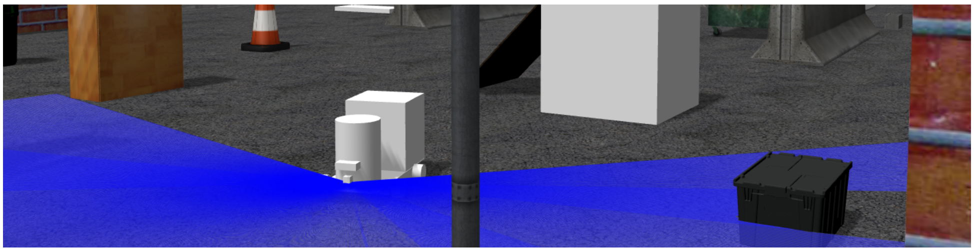


Figure 5.1: The "Asphalt" world in Gazebo.

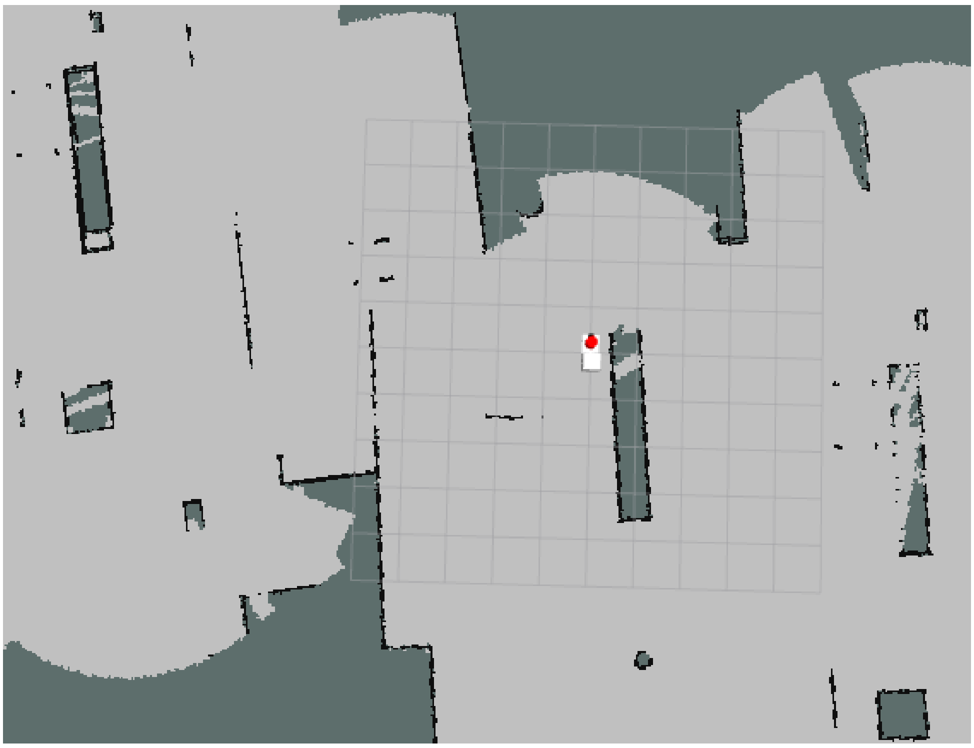


Figure 5.2: An example of incorrect map merging. This case occurred in the "Asphalt" world simulated in Gazebo.

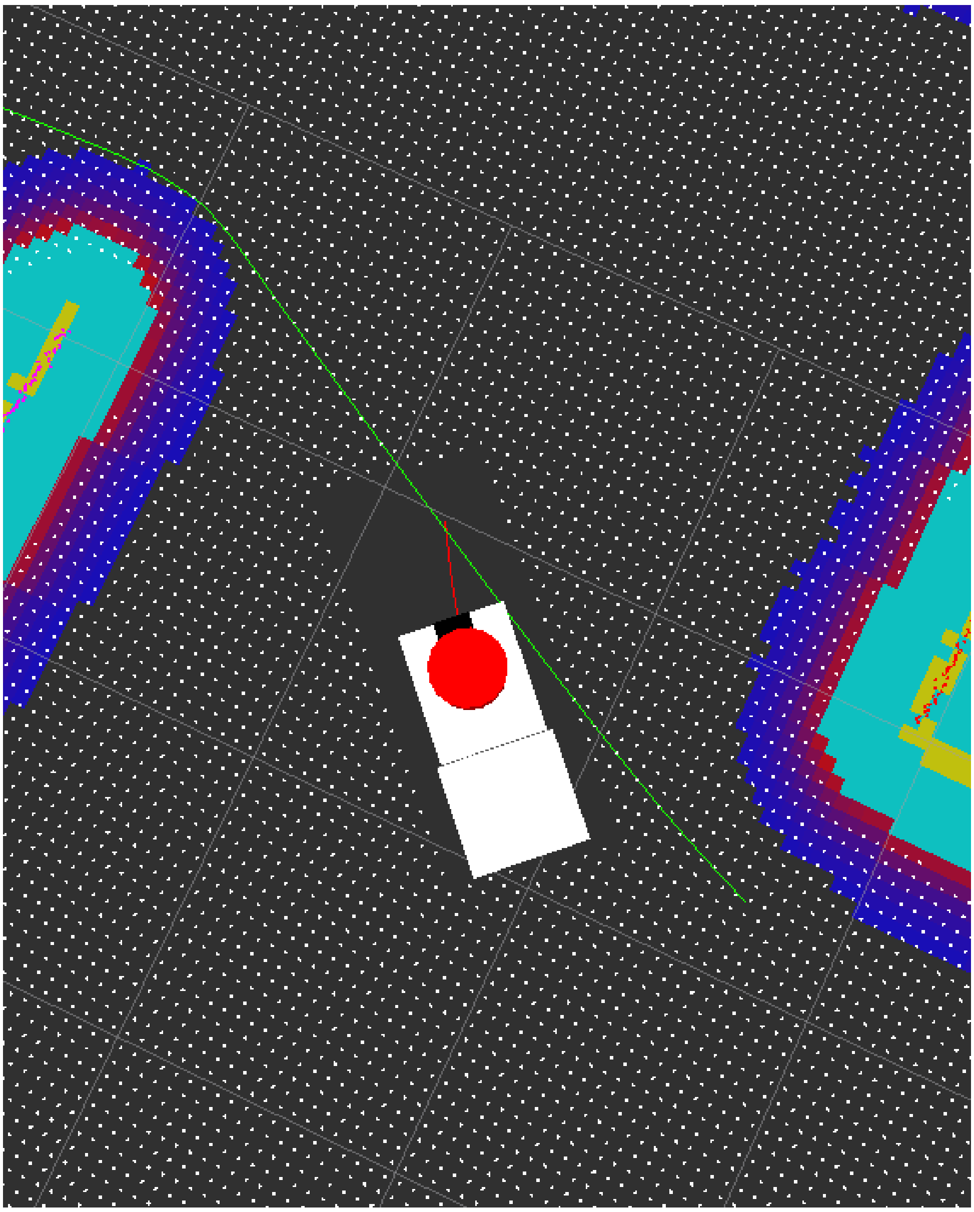


Figure 5.3: Nodes and topics for motion control.

Chapter 6

Discussion

Chapter 7
Conclusion

7.1 Future Work

7.1.1 Autonomous Non-Destructive Testing

Advancements in Artificial Intelligence (AI), big data and machine learning opens up exciting possibilities for autonomous Non-destructive Testing (NDT). Branches of this technology is usually encountered in the context of image recognition, i.e. teaching machines to understand what they see. The same concepts may be applied to forms of NDT besides regular visual sensor input, such as ultra sound or eddy currents for corrosion detection.

7.1.2 Large Scale Kinect Fusion - Kintinous

Kinect Fusion has great potential for augmented reality. Augmented reality is a concept which blends the real and virtual environment. This opens up opportunities to create realistic and immersive training scenarios for the operators. Unfortunately, Kinect Fusion is limited reconstructing a rather small volume depending on the resolution. By varying the resolution, volumes can at the least cover a normal office desk and at the most cover a small room [**?**].

Kintinous...

A guide on how to build Kintinous can be found at https://github.com/mp3guy/Kintinuous. The procedure is complicated, as it usually is for experimental builds. It is recommended to attempt the procedure on a fresh install of Ubuntu 14.04 or 15.04 [Kin].

Improve the Communication Protocols

Communication between ROS and the XMEGA A3BU, the Bluetooth device and the OCS, all use the same pattern: A start byte `":"`, the message with the speed setting and a stop byte `"Esc"`. In later projects, it could be beneficial to implement a more robust and rich communication protocol with more options for remote operation.

Implement a Fully Functional Operator Control Station

At the end of this project, the OCS provided functionality for moving the robot, and displaying live video from the Kinect.

7.1.3 Hardware

Several hardware-related issues became apparent over the course of the project - especially toward the final weeks. These issues are likely the results of many disconnected projects on the same hardware.

Kinect Sensor Location

This is the first semester in which a Kinect has been used on the robot. At the moment, the sensor is placed directly over the LIght Detection And Ranging (LIDAR). Because the depth sensor in the Kinect for XBOX 360 has a minimum range of roughly $0.5m$, it cannot detect objects within reach of the robot arm. It is recommended to find a new location further back on the robot.

Combine Stereo Cameras with Kinect-like Sensors

As mentioned, both active and passive depth cameras have limitations. The Kinect does function in direct sunlight, but it can measure depth in the dark. Passive depth sensors, for example stereo cameras, does depend on visible light to sense anything at all. While RTAB-Map does depend on visibility for loop closure detection, there are other SLAM methods, e.g. *Kinect Fusion*, which do not. An implementation could use a light sensor to sense light that may interfere with the Kinect. Light levels could be compared to a threshold and switch between the stereo cameras or the Kinect depending on how well each sensor will work in the current conditions.

On-board Computer Suitable for Moving Platforms

Because this author used his own computer to control the robot, all features related to ROS was removed from the robot at the end of the project. A new computer should be equipped with Solid State Drive (SSD) storage

Wheels

There were mainly two issues with the omni-wheels this semester: They are worn out, and one wheel slipped out of

the motor drive shaft. The rubber on a few of the perpendicular rollers is either loose or about to fall off the plastic rims. This causes the robot to shake, which can damage spinning hard disk drives or shake the sensors out of their calibrated positions.

7.2 Task Fulfillment

7.3 Task Fulfilment

7.4 Final Conclusion

References

[Bog16] Robert Bogue. Europe continues to lead the way in the collaborative robot business. *Industrial Robot: An International Journal*, 43(1):6–11, 2016.

[DRC] Darpa robotics challenge. http://www.theroboticschallenge.org/overview. Accessed: 2016-04-05.

[ER12] Mohamed A. El-Reedy. Chapter 6 - corrosion protection. In Mohamed A. El-Reedy, editor, *Offshore Structures*, pages 383 – 443. Gulf Professional Publishing, Boston, 2012.

[GC11] Peter Gorle and Andrew Clive. Positive impact of industrial robots on employment. Report, METRA MARTECH Limited, 2011.

[ifr] International federation of robotics - statistics. http://www.ifr.org/industrial-robots/statistics/. Accessed: 2016-04-05.

[JWA$^+$12] J. Jamieson, L. Wilson, M. Arredondo, K. Evans, J.and Hamilton, and C Sotzing. Autonomous inspection vehicle: A new dimension in life of field operations. *Offshore Technology Conference*, April 2012.

[KFO12] Shinji Kawatsuma, Mineo Fukushima, and Takashi Okada. Emergency response by robots to fukushima daiichi accident: summary and lessons learned. *Industrial Robot: An International Journal*, 39(5):428–435, 2012.

[Kin] Kintinous. https://github.com/mp3guy/Kintinuous. Accessed: 2016-03-21.

[KMP15] K. Kydd, S. Macrez, and P. Pourcel. *Autonomous Robot for Gas and Oil Sites*. Society of Petroleum Engineers, September 2015.

[LM14] M. Labbé and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2661–2666, Sept 2014.

[MIM] MIMROex, mobile maintenance and inspection robot for process plants. http://www.ipa.fraunhofer.de/fileadmin/user_upload/Kompetenzen/Roboter-_und_Assistenzsysteme/Industrielle_und_gewerbliche_Servicerobotik/English_Documents/Product_sheet_MIMROex_Mobile_maintenance_and_inspection_robot_for_process_plants.pdf.

[pre] Press release: Subsea 7 completes design and build of first commercial autonomous inspection vehicle (aiv). http://www.subsea7.com/content/dam/subsea7/Company%20News/2011/Subsea7completesdesignandbuildoffirstcommericalAIV.pdf. Accessed: 2016-04-13.

[QGS15] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS*. O'Reilly Media, Inc., December 2015.

[RCR$^+$15] Pere Ridao, Marc Carreras, David Ribas, Pedro J. Sanz, and Gabriel Oliver. Intervention auvs: The next challenge. *Annual Reviews in Control*, 40:227 – 241, 2015.

[Reb] Reboot ubuntu. http://askubuntu.com/questions/4408/what-should-i-do-when-ubuntu-freezes/36717#36717. Accessed: 2016-03-14.

[ROSa] ROS history. http://www.ros.org/history/. Accessed: 2016-02-28.

[ROSb] ROS installation. http://wiki.ros.org/indigo/Installation/Ubuntu. Accessed: 2016-02-29.

[ROSc] ROS on the iss. http://www.ros.org/news/2014/09/ros-running-on-iss.html. Accessed: 2016-04-10.

[SWB$^+$14] Samuel Soldan, Jochen Welle, Thomas Barz, Andreas Kroll, and Dirk Schulz. Towards autonomous robotic systems for remote gas leak detection and localization in industrial environments. In *Field and Service Robotics*, pages 233–247. Springer Berlin Heidelberg, 2014.

[WA12] Jarrett Webb and James Ashley. *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012.

Appendix A
Setting Up the Project

A.1 Installation

A.1.1 Equipment List

Item List

Software list

Hector SLAM for ROS Install with

```
sudo apt-get install ros-indigo-hector-slam
```

Compatibility Issues

Indigo, Ubuntu etc.

A.1.2 Install Ubuntu

A.1.3 Download ROS

A.2 Configuring the Project

A.2.1 Configuring the ROS Workspace

A.2.2 Configuring the Bluetooth Connection

The Qt framework is used to simplify the implementation of the Bluetooth connection between the ROS graph and a remote device. Our ROS installation for this project already includes some variant of Qt version 4.8. While useful for creating new GUI applications, it lacks a Bluetooth API. The latest version of Qt, version 5.x, is equipped with libraries necessary for developing Bluetooth applications. This part of the guide will explain how to create a Qt 5 application which can be build by *catkin_make* and run as a *rosnode*.

1 - Install Qt5

Installing Qt5 for Linux is a straight forward procedure. Go to qt.io, and download the free version of Qt. All necessary instructions are provided. Qt5 may be installed in the home folder.

2- Enabling Qt5 in a ROS node

It is assumed that the ROS package ”bluetooth_server”, is located in a catkin workspace:

```
<NAME OF CATKIN WORKSPACE>/src/bluetooth_server
```

Inside this folder, open the file ”CMakeLists.txt” and locate the following:

```
set(CMAKE_PREFIX_PATH "/home/vegard/Qt/5.5/gcc_64/lib/cmake/Qt5"
  "/home/vegard/Qt/5.5/gcc_64/lib/cmake/Qt5Core"
  "/home/vegard/Qt/5.5/gcc_64/lib/cmake/Qt5Bluetooth"
```

Change these paths to the correct paths on your system.

Appendix B
Troubleshooting

B.1 Introduction

This chapter contains proposed solutions to some of the problems that was encountered over the course od the semester. The solutions are not complete or comprehensive, but may provide some quick fixes for any students that may continue working with this project.

B.2 ROS

B.2.1 ERROR: tf2:ExtrapolationException

When running the released ROS distribution binary of `RTAB-Map` installed with `apt-get` on the robot, the node would crash after a few iterations. The error message is as follows:

```
Lookup would require extrapolation into the future, ..., when looking
up transform from frame [laser] to frame [base_link].
```

The requested transform is milliseconds ahead of ”now”. This issue was fixed by maintainers in March 2016[1], but was not yet integrated into the released binary. During work with this project, the problem was solved by building `RTAB-Map` from source, where the most recent fixes are included. This is a straight forward procedure, which is described on the project's GitHub repository.

[1]https://github.com/introlab/rtabmap_ros/issues/54

B.3 Gazebo

B.3.1 Error [Node.cc.90] No namespace found

Solution: Remember to source the *gazebo* installation. In this case, with *gazebo-2.2* installed as recommended for ROS Indigo, the setup file can be sourced by typing

```
$ source /usr/share/gazebo-2.2/setup.sh
```

B.3.2 Dependency Issues When Installing *gazebo2*

This problem was encountered after removing gazebo and then typing

```
$ sudo apt-get upgrade
```

When typing

```
$ sudo apt-get install -y gazebo2
```

the installation failed because some dependencies had been upgraded to an incompatible version. To solve this, take note of the missing dependencies listed after entering the command above, open Ubuntu Software Center and select the History tab. Scroll down and locate the missing dependencies. They should have a red X next to them, indicating that they have been uninstalled. Then, enter the following command:

```
$ sudo apt-get install <NAME OF THE UNINSTALLED DEPENDENCY>
```

gazebo2

B.4 Ubuntu

B.4.1 Ubuntu Freezes

Sometimes during work with the project, Ubuntu would freeze and become unresponsive to keyboard input and mouse clicks. The mouse could be moved around, but was otherwise unresponsive. This event occurred exclusively when using *RVIZ* and displaying a camera topic as an image in the lower left corner of the GUI. The following steps from a post at askubuntu.com, solves the problem [Reb]:

While holding Alt and SysReq (Print Screen) , type R E I S U B . Press each key properly, and allow a few seconds to pass between each keystroke so that each command has time to execute. This should cause the computer to reboot, and is supposedly safer than using the power button. Follow **this** link to the post for more details.